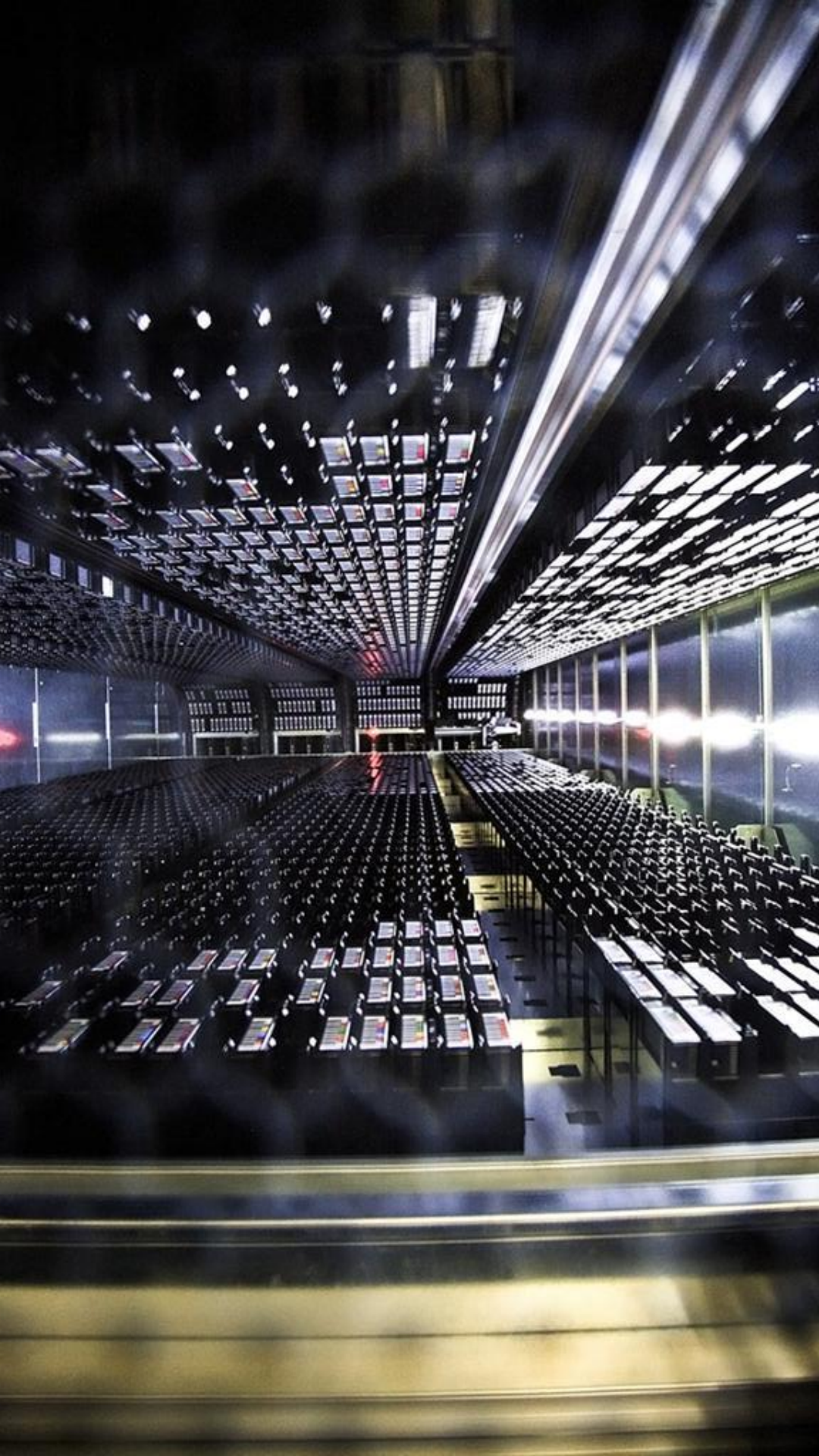

CERN

Welcome to the CERN Data Centre

.. and an introduction to one aspect of Scientific Computing

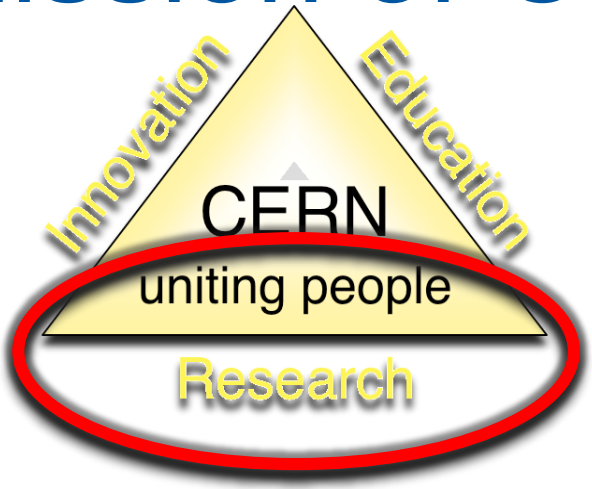

StorageTek
StorageTek



Scientific computing

With an highlight on storage …

The mission of CERN



We are here

Accelerating particle beams

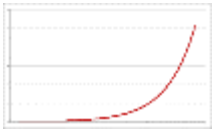
Detecting particles (experiments)

Large-scale computing (Analysis)

Discovery

The need for computing in research

- Scientific research in recent years has exploded the computing requirements
  - Computing has been the strategy to reduce the cost of traditional research



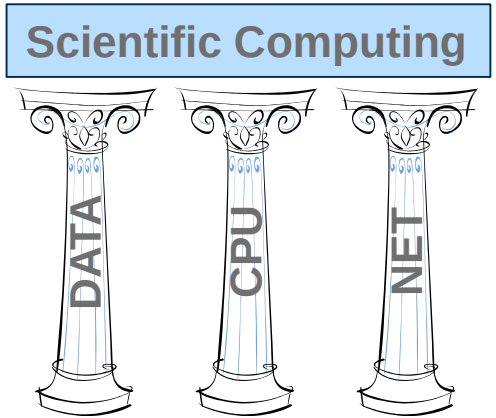
At constant cost, exponential growth of performances

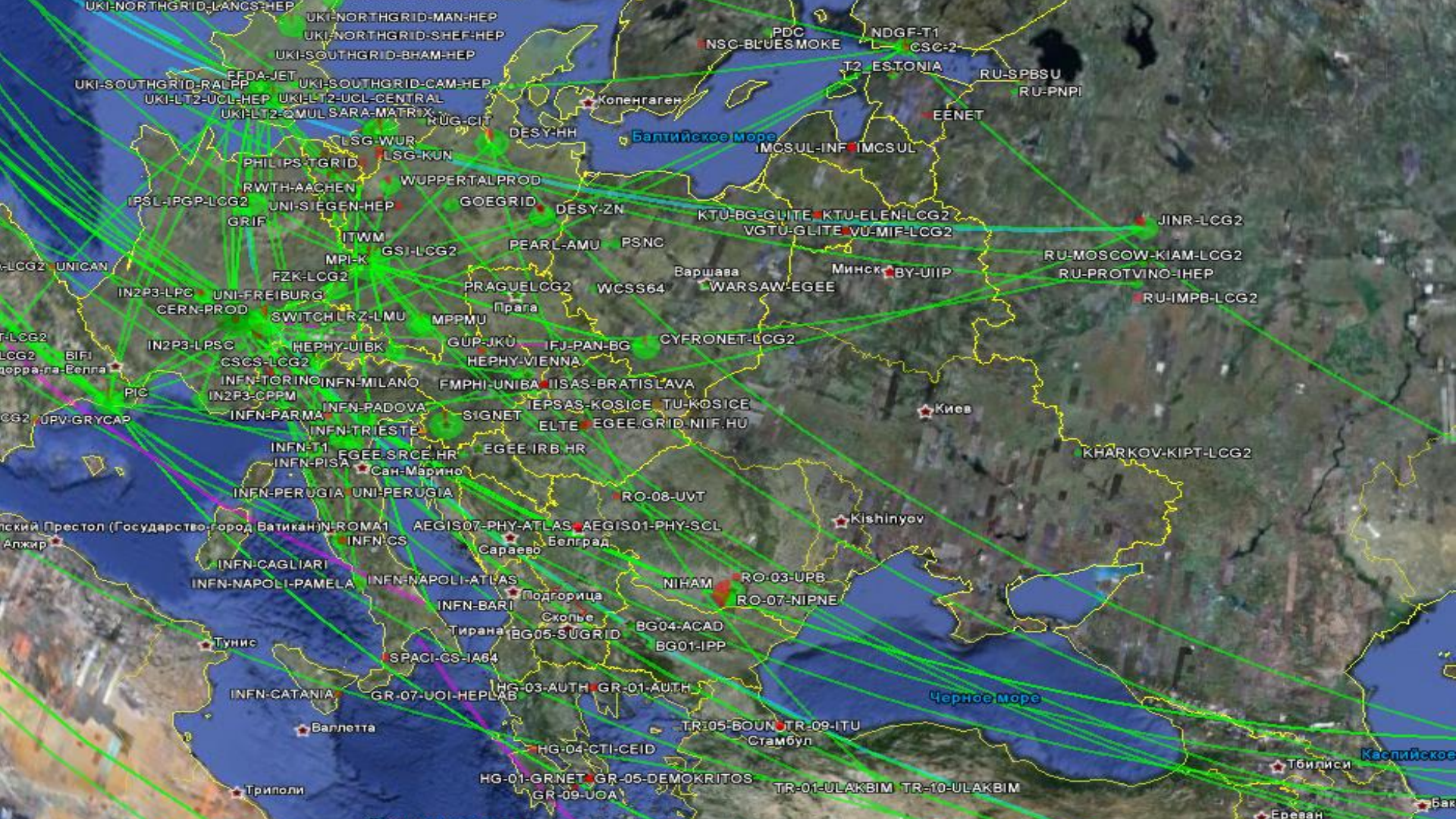
  - Computing has opened new horizons of research not only in High Energy Physics

Return in computing investment higher than other fields: Budget available for computing increased, growth is more than exponential

The need for storage in computing

- Scientific computing for large experiments is typically based on a distributed infrastructure
- Storage is one of the main pillars
- Storage requires Data Management…



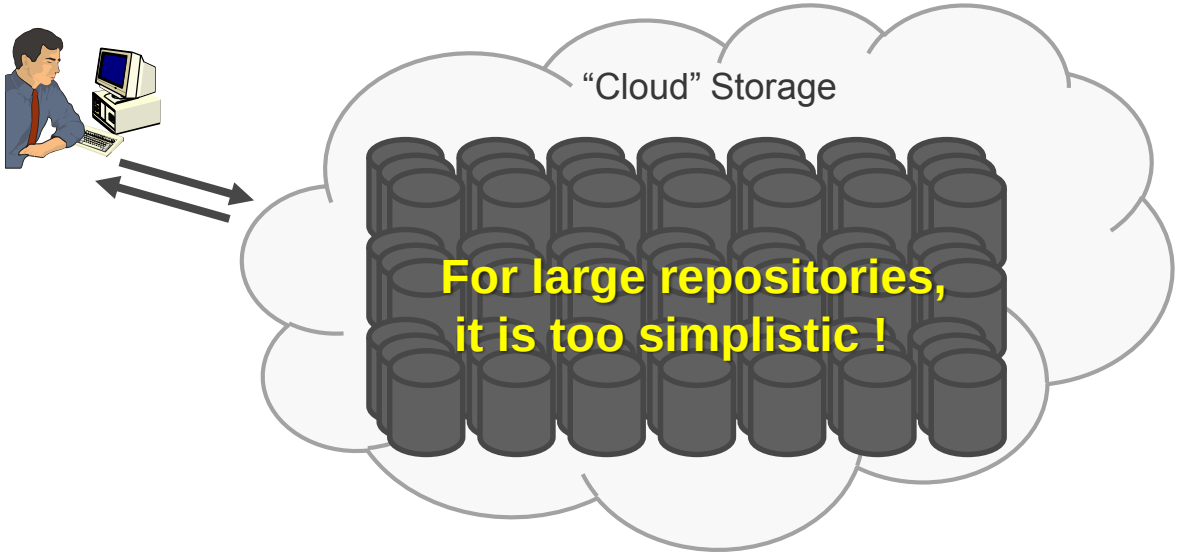

UKI-NORTHGRID-LANCS-HEP
UKI-NORTHGRID-MAN-HEP
UKI-NORTHGRID-SHEF-HEP
UKI-SOUTHGRID-BHAM-HEP
UKI-SOUTHGRID-RALPP
UKI-SOUTHGRID-CAM-HEP
UKI-LT2-UCL-HEP
UKI-LT2-UCL-CENTRAL
UKI-LT2-QMUL
SARA-MATRIX
RUG-CIT
LSG-WUR
LSG-KUN
PHILIPS-TGRID
RWTH-AACHEN
WUPPERTALPROD
IPSL-IPGP-LCG2
UNI-SIEGEN-HEP
GOEGRID
DESY-ZN
DESY-HH
GRIF
ITWM
GSI-LCG2
MPI-K
PEARL-AMU
PSNC
FZK-LCG2
UNICAN
IN2P3-LPC
UNI-FREIBURG
CERN-PROD
SWITCH
LRZ-LMU
MPPMU
PRAGUELCG2
WCSS64
WARSAW-EGEE
IN2P3-LPSC
HEPHY-UIBK
GUP-JKU
IFJ-PAN-BG
CYFRONET-LCG2
HEPHY-VIENNA
CSCS-LCG2
INFN-TORINO
INFN-MILANO
FMPHI-UNIBA
ISAS-BRATISLAVA
PIC
IN2P3-CPPM
INFN-PADOVA
EPSAS-KOSICE
TU-KOSICE
INFN-PARMA
SIGNET
ELTE
EGEE.GRID.NIIF.HU
UPV-GRYCAP
INFN-TRIESTE
INFN-T1
EGEE.SRCE.HR
EGEE.IRB.HR
INFN-PISA
INFN-PERUGIA
UNI-PERUGIA
RO-08-UVT
INFN-ROMA1
AEGIS07-PHY-ATLAS
AEGIS01-PHY-SCL
INFN-CS
INFN-CAGLIARI
INFN-NAPOLI-PAMELA
INFN-NAPOLI-ATLAS
NIHAM
RO-03-UPB
RO-07-NIPNE
INFN-BARI
BG05-SUGRID
BG04-ACAD
BG01-IPP
SPACI-CS-IA64
INFN-CATANIA
GR-07-UOI-HEPLAB
HG-03-AUTH
GR-01-AUTH
HG-04-CTI-CEID
HG-01-GRNET
GR-05-DEMOKRITOS
GR-09-UOA
TR-05-BOUN
TR-09-ITU
TR-01-ULAKBIM
TR-10-ULAKBIM
PDC
NSC-BLUESMOKE
NDGF-T1
CSC-2
T2_ESTONIA
RU-SPBSU
RU-PNPI
EENET
IMCSUL-INF
IMCSUL
KTU-BG-GLITE
KTU-ELEN-LCG2
VGTU-GLITE
VU-MIF-LCG2
JINR-LCG2
RU-MOSCOW-KIAM-LCG2
RU-PROTVINO-IHEP
RU-IMPB-LCG2
BY-UIIP
KHARKOV-KIPT-LCG2
Копенгаген
Балтийское море
Варшава
Минск
Прага
Киев
Kishinyov
Сараево
Белград
Подгорица
Скопье
Тирана
Тунис
Валлетта
Триполи
Стамбул
Черное море
Каспийское
Тбилиси
Ереван
Сан-Марино
Алжир

“Why” data management ?

- Data Management solves the following problems
  - Data reliability
  - Access control
  - Data distribution
  - Data archives, history, long term preservation
  - In general:
    - Empower the implementation of a workflow for data processing

Can we make it simple ?

- A simple storage model: all data into the same container
  - Uniform, simple, easy to manage, no need to move data
  - Can provide sufficient level of performance and reliability

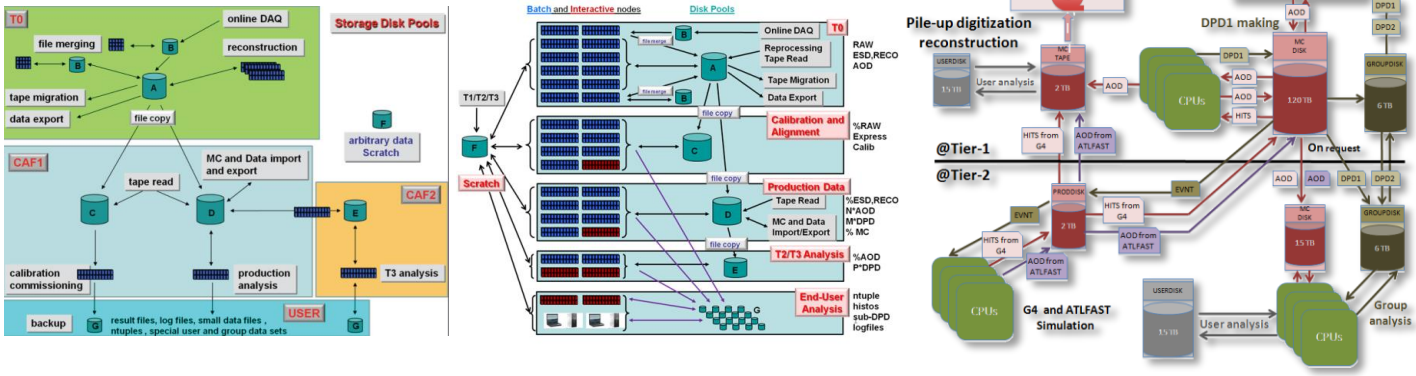


Why multiple pools and quality ?

- Derived data used for analysis and accessed by thousands of nodes
  - Need high performance, Low cost, minimal reliability (derived data can be recalculated)
- Raw data that need to be analyzed
  - Need high performance, High reliability, can be expensive (small sizes)
- Raw data that has been analyzed and archived
  - Must be low cost (huge volumes), High reliability (must be preserved), performance not necessary

So, … what is data management ?

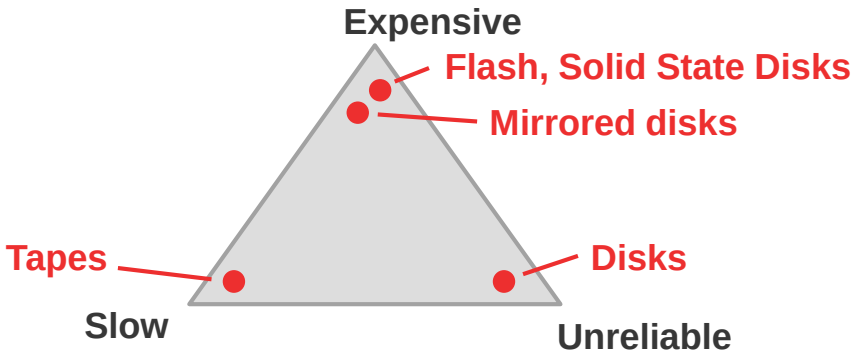
- Examples from LHC experiment data models



- **Two building blocks to empower data processing**
  - Data pools with different quality of services
  - Tools for data transfer between pools

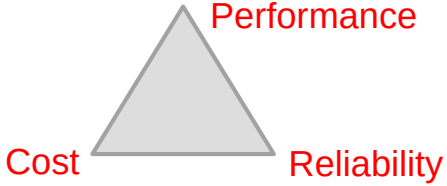
Data pools

- Different quality of services
  - Three parameters: (Performance, Reliability, Cost)
  - You can have two but not three

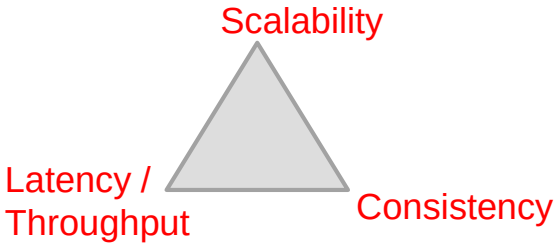


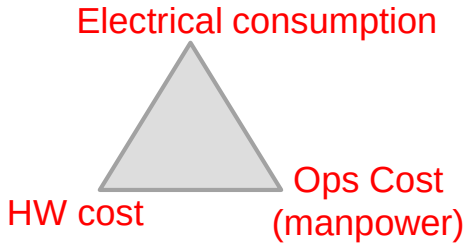
But the balance is not as simple

- Many ways to split (performance, reliability, cost)



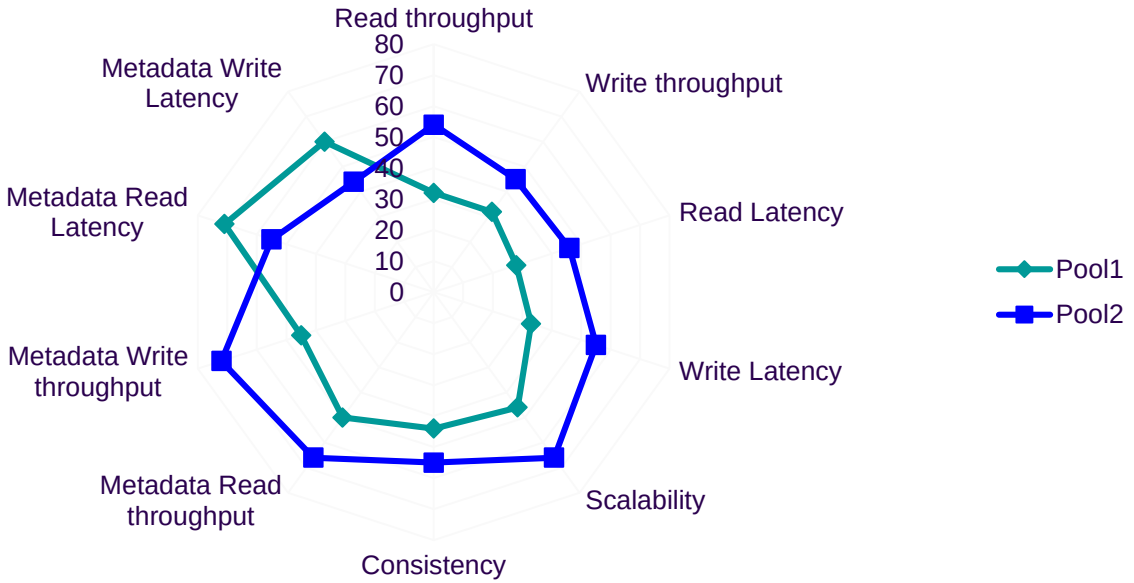
- Performance has many sub-parameters
- Cost has many sub-parameters
- Reliability has many sub-parameters





And reality is complicated

- Key requirements: Simple, Scalable, Consistent, Reliable, Available, Manageable, Flexible, Performing, Cheap, Secure.
- Aiming for “à la carte” services (storage pools) with on-demand “quality of service”



Areas of research in Storage

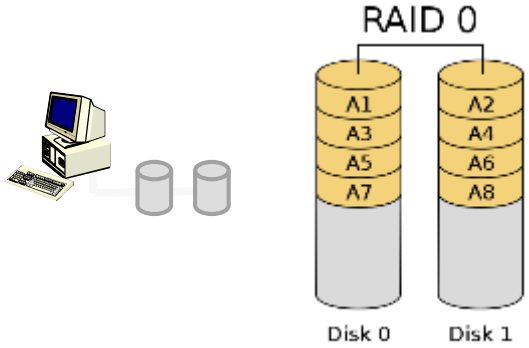
Reliability, Scalability, Security, Manageability

Storage Reliability

- Reliability is related to the probability to lose data
  - Def: “the probability that a storage device will perform an arbitrarily large number of I/O operations without data loss during a specified period of time”
- Reliability of the “service” starts from the reliability of the underlying hardware
  - Example of disk servers with simple disks: reliability of service = reliability of disks
- But data management solutions can increase the reliability of the hardware at the expenses of performance and/or additional hardware / software
  - Redundant Array of Inexpensive Disks (RAID)

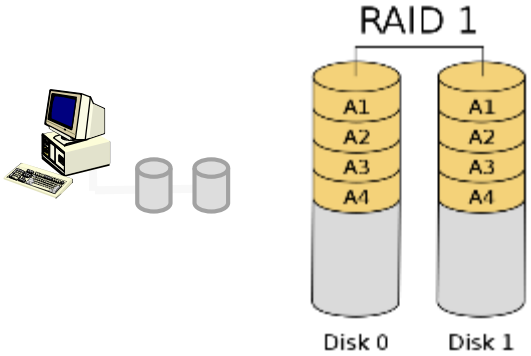
Reminder: types of RAID

- RAID0
  - Disk striping
- RAID1
  - Disk mirroring
- RAID5
  - Parity information is distributed across all disks
- RAID6
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss

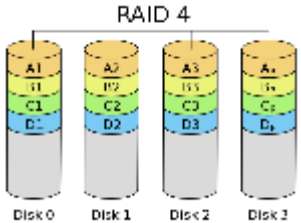


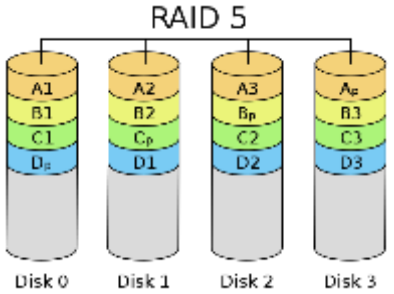
Reminder: types of RAID

- RAID0
  - Disk striping
- RAID1
  - Disk mirroring
- RAID5
  - Parity information is distributed across all disks
- RAID6
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss



Reminder: types of RAID



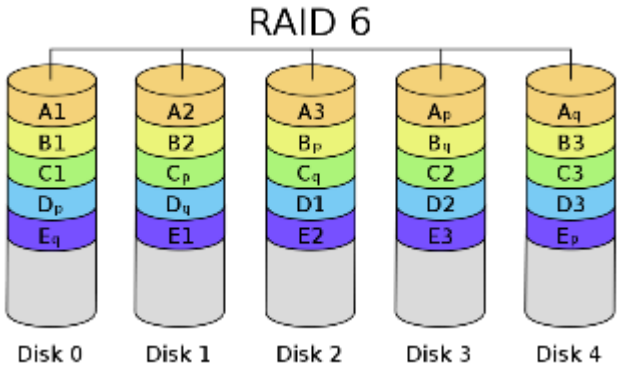


- RAID0
  - Disk striping
- RAID1
  - Disk mirroring
- RAID5
  - Parity information is distributed across all disks
- RAID6
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss

http://en.wikipedia.org/wiki/RAID

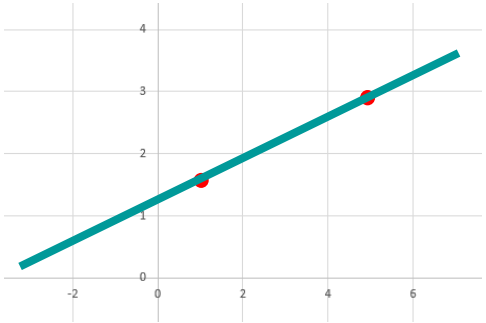
Reminder: types of RAID

- RAID0
  - Disk striping
- RAID1
  - Disk mirroring
- RAID5
  - Parity information is distributed across all disks
- RAID6
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss

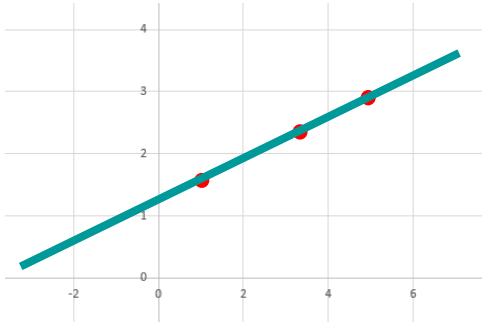


Understanding error correction

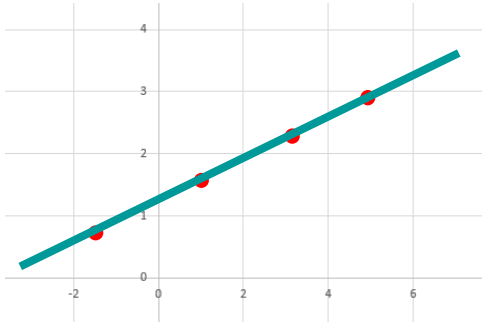
- A line is defined by 2 numbers: a, b
  - (a, b) is the information
  - y = ax + b
- Instead of transmitting a and b, transmit some points on the line at known abscissa. 2 points define a line. If I transmit more points, these should be aligned.



2 points



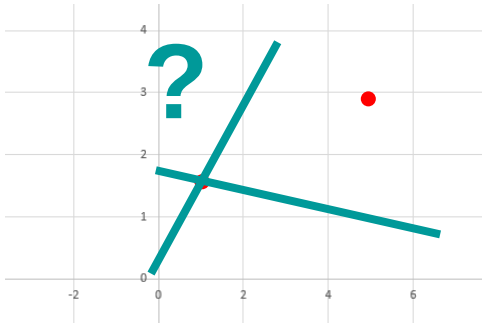
3 points



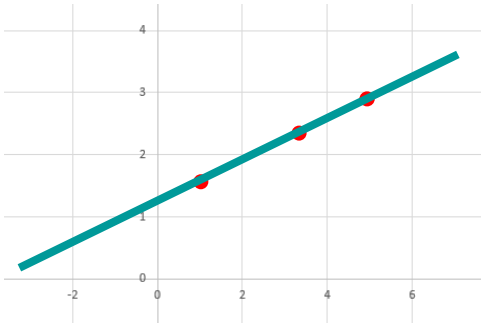
4 points

If we lose some information …

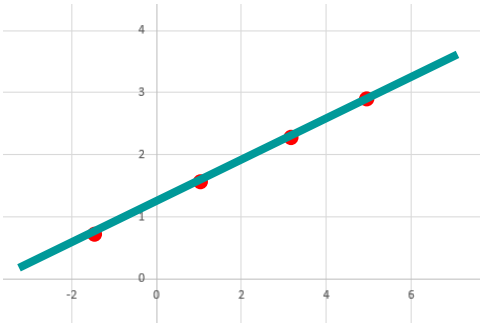
- If we transmit more than 2 points, we can lose any point, provided the tot`al number of point left is >= 2



1 point instead of 2
information lost



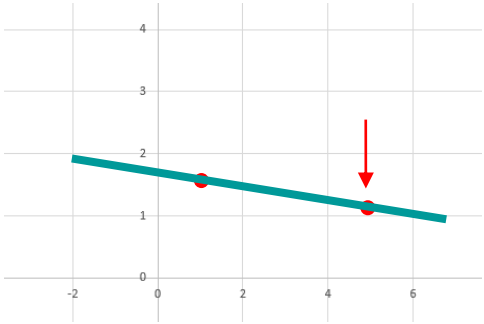
2 points instead of 3



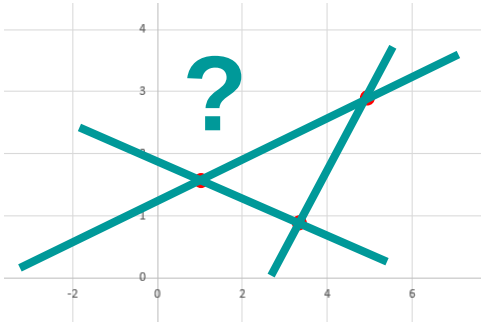
2 or 3 points instead of 4

If we have an error …

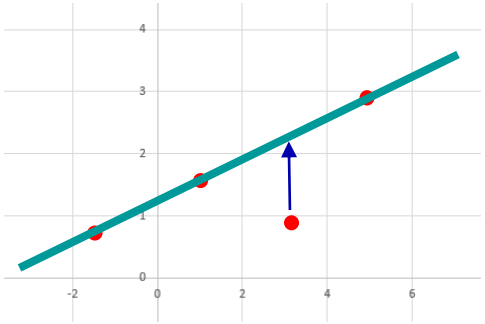
- If there is an error, I can detect it if I have transmitted more than 2 points, and correct it if have transmitted more than 3 points



Information lost
(and you do not notice)



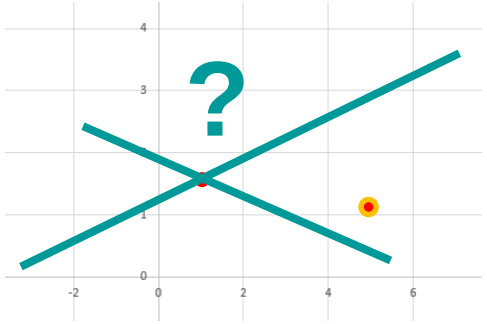
Error detection
Information is lost
(and you notice)



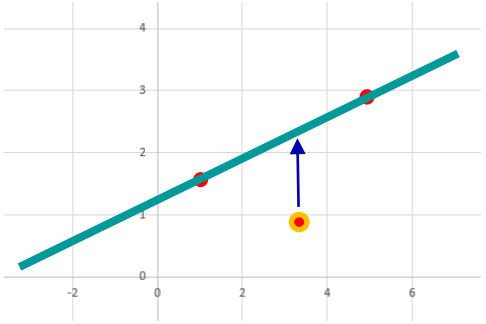
Error correction
Information is recovered

If you have checksumming on data …

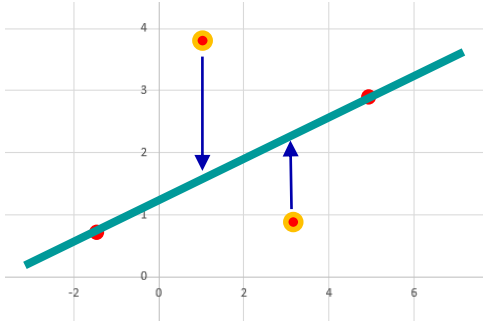
- You can detect errors by verifying the consistency of the data with the respective checksums. So you can detect errors independently.
- … and use all redundancy for error correction



Information lost
(and you notice)



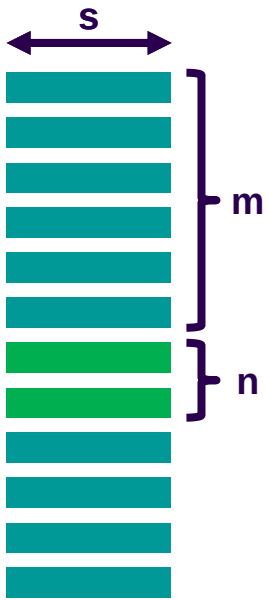
Error correction
Information is recovered



2 Error corrections possible
Information is recovered

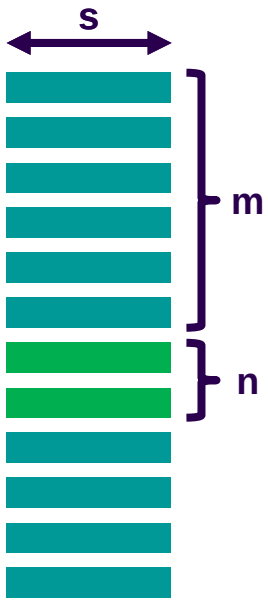
Arbitrary reliability

- RAID is “disks” based. This lacks of granularity
- For increased flexibility, an alternative would be to use files ... but files do not have constant size
- File “chunks” (or “blocks”) is the solution
  - Split files in chunks of size “s”
  - Group them in sets of “m” chunks
  - For each group of “m” chunks, generate “n” additional chunks so that
    - For any set of “m” chunks chosen among the “m+n” you can reconstruct the missing “n” chunks
  - Scatter the “m+n” chunks on independent storage



Arbitrary reliability with the “chunk” based solution

- The reliability is independent form the size “s” which is arbitrary.
  - Note: both large and small “s” impact performance
- Whatever the reliability of the hardware is, the system is immune to the loss of “n” simultaneous failures from pools of “m+n” storage chunks
  - Both “m” and “n” are arbitrary. Therefore arbitrary reliability can be achieved
- The fraction of raw storage space loss is n / (n + m)
- Note that space loss can also be reduced arbitrarily by increasing m
  - At the cost of increasing the amount of data loss if this would ever happen



Analogy with the gambling world

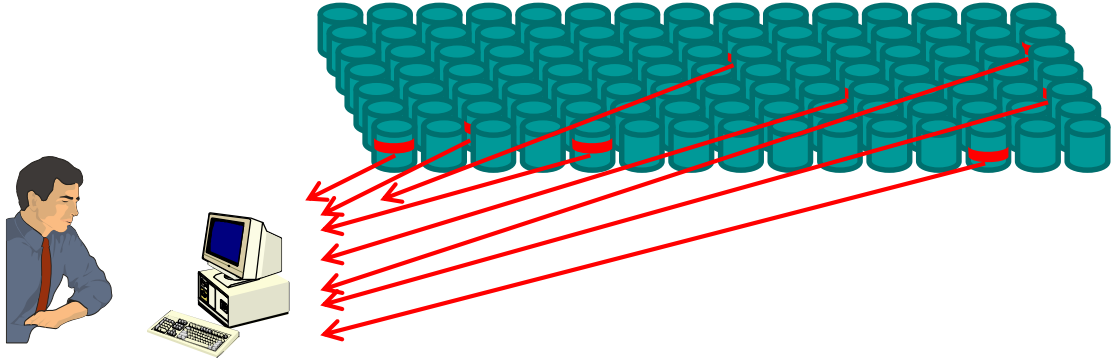
- We just demonstrated that you can achieve “arbitrary reliability” at the cost of an “arbitrary low” amount of disk space. This is possible because you increase the amount of data you accept loosing when this rare event happens.
- In the gambling world there are several playing schemes that allows you to win an arbitrary amount of money with an arbitrary probability.
- Example: you can easily win 100 Euros at > 99 % probability ...
  - By playing up to 7 times on the “Red” of a French Roulette and doubling the bet until you win.
  - The probability of not having a “Red” for 7 times is (19/37)7 = 0.0094)
  - You just need to take the risk of loosing 12’700 euros with a 0.94 % probability

Amount		Win		Lost	
Bet	Cumulated	Probability	Amount	Probability	Amount
100	100	48.65%	100	51.35%	100
200	300	73.63%	100	26.37%	300
400	700	86.46%	100	13.54%	700
800	1500	93.05%	100	6.95%	1500
1600	3100	96.43%	100	3.57%	3100
3200	6300	98.17%	100	1.83%	6300
6400	12700	99.06%	100	0.94%	12700

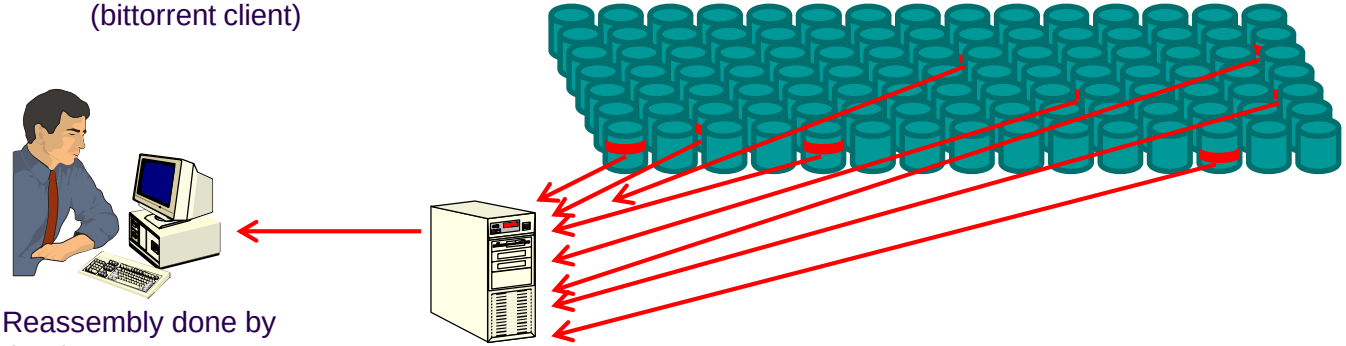
Chunk transfers

- Among many protocols, Bittorrent is the most popular
- An SHA1 hash (160 bit digest) is created for each chunk
- All digests are assembled in a “torrent file” with all relevant metadata information
- Torrent files are published and registered with a tracker which maintains lists of the clients currently sharing the torrent’s chunks
- In particular, torrent files have:
  - an "announce" section, which specifies the URL of the tracker
  - an "info" section, containing (suggested) names for the files, their lengths, the list of SHA-1 digests
- Reminder: it is the client’s duty to reassemble the initial file and therefore it is the client that always verifies the integrity of the data received

Reassembling the chunks



Data reassembled directly on the client (bittorrent client)



Reassembly done by the data management infrastructure

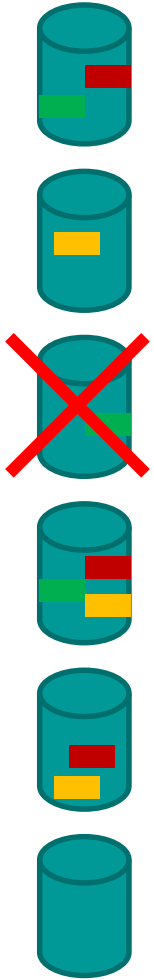
Middleware

Example: High Availability with replication

- We have “sets” of T independent storage
  - This example has T=6
- The storage pool is configured to replicate files R times, with R < T
  - This example: R=3 every file is written 3 times on 3 independent storage out of the 6 available
  - When a client read a file, any copy can be used
  - Load can be spread across the multiple servers to ensure high throughput (better than mirrored disks, and much better than Raid 5 or Raid 6)

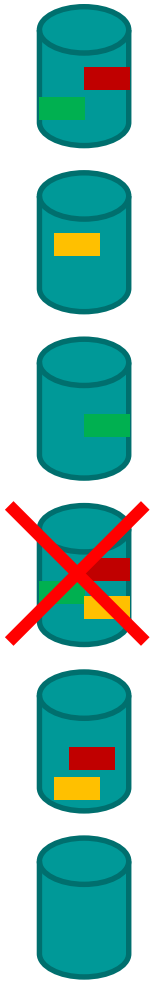
Example scenario: hardware failure

- The loss of a storage component is detected. The storage component is disabled automatically
- File Read requests can continue if R>1 (at least 1 replica), at reduced throughput
  - The example has R=3
- File Creation / Write requests can continue
  - New files will be written to the remaining T – 1 = 6 – 1 = 5 storage components
- File Delete request can continue
- File Write / Update requests can continue
  - Either by just modifying the remaining replicas or by creating on the fly the missing replica on another storage component
- Service operation continues despite hardware failure. (remember: independent storage)



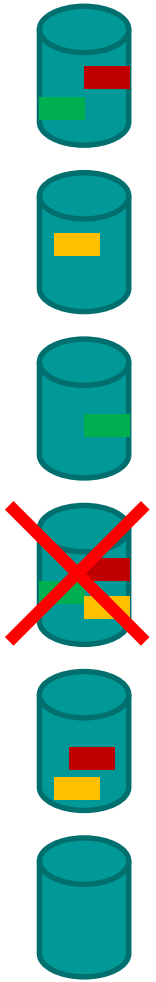
Example scenario: failure response

- The disabled faulty storage is not used anymore
- There is “Spare Storage” that can be used to replace faulty storage
  - manually or automatically
- The lost replicas are regenerated from the existing replicas
  - Manually or automatically



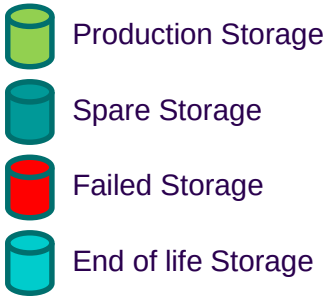
Example scenario: draining a server

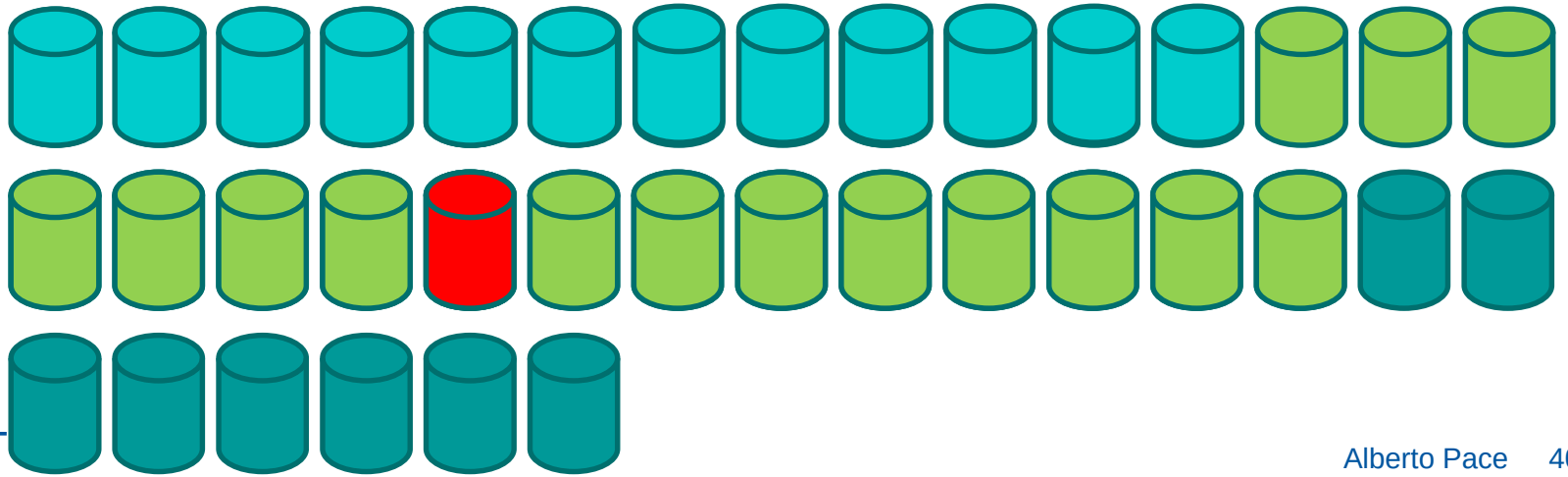
- To drain a server, just power it off
- Will be seen as faulty and disabled (it will not used anymore)
- The available “Spare Storage” will be used to replace faulty storage
  - manually or automatically
- The lost replicas are regenerated from the existing replicas
  - Manually or automatically



Service operation eased …

- Production cluster, 15 Server with 9 spare
- Server Failure (servers)
- New HW delivery (6 servers)Out of warranty (6 servers)
- End of life





Roles Storage Services

- Three main roles
  - Storage (store the data)
  - Distribution (ensure that data is accessible)
  - Preservation (ensure that data is not lost)

Size in PB + performance

Availability

Reliability

CERN Computing Infrastructure

COMPUTING		31-Oct-2017 @ 19:55	STORAGE	NETWORK	
Servers (Meyrin)	Cores (Meyrin)	Disks (Meyrin)	Tape Drives	Routers	Star Points
11.1 K	166 K	61.3 K	91	236	679
Servers (Wigner)	Cores (Wigner)	Disks (Wigner)	Tape Cartridges	Switches	Wifi Points
3.5 K	56.0 K	29.7 K	26.7 K	3.9 K	1.7 K

CPUs Network Databases Storage Infrastructure

www.cern.ch